TECHNICAL REPORT
NATICK/TR-07/004

AD_____

AGENT-BASED MODELING OF DISMOUNTED INFANTRY THROUGH INCLUSION OF PERCEPTIONS, INFERENCES AND ASSOCIATIONS PHASE I

by
Walter Warwick

Micro Analysis and Design
Boulder, CO 80301

November 2006

Final Report
November 2005 – May 2006

Approved for public release; distribution is unlimited

Prepared for
U.S. Army Research, Development and Engineering Command
Natick Soldier Center
Natick, Massachusetts 01760-5020

DISCLAIMERS

The findings contained in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of trade names in this report does not constitute an official endorsement or approval of the use of such items.

DESTRUCTION NOTICE

For Classified Documents:

Follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For Unclassified/Limited Distribution Documents:

Destroy by any method that prevents disclosure of contents or reconstruction of the document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
07-11-2006	Final	November 2005 - May 2006

4. TITLE AND SUBTITLE
AGENT-BASED MODELING OF DISMOUNTED INFANTRY THROUGH INCLUSION OF PERCEPTIONS, INFERENCES AND ASSOCIATIONS PHASE I

5a. CONTRACT NUMBER
W911QY-06-C-0005

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
Clin 0001

6. AUTHOR(S)
Walter Warwick

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Micro Analysis and Design
4949 Pearl East Circle, Suite 300
Boulder, CO 80301

8. PERFORMING ORGANIZATION REPORT NUMBER
Data Item A004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Soldier Systems Center
Modeling & Analysis, ATTN: AMSRD-NSC-SS-MA
Kansas St., Natick, MA 01760-5020

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NATICK/TR-07/004

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Report developed under Small Business Innovation Research contract. Where human behavior is often thought of in terms of a perception-action cycle, rich with interdependencies and fuzzy boundaries between processes, human behavior representations of a computer-generated force implement a one-way process where exact, unambiguous data drive discrete monolithic models of decision-making.The objective was to imbue human behavior representations with some complexities that characterize human perception and action. In more specific terms, Micro Analysis and Design's Phase I effort revolved around two related objectives. The first was to design and prototype a "Perceptual Editor" that could be integrated within Infantry Warrior Simulation (IWARS) environment and allow a user to define different kinds of transformations between ground truth variables and agent-based perceptions.The second was to investigate examples of the kinds of transformations that might be implemented using such an editor. We describe progress toward both objectives. We begin with a discussion of software developed to implement our "Perceptual Editor." We then characterize performance of two different mechanisms for transforming ground truth into"perceptions." We conclude with some general methodological remarks about the process of integrating perceptual inferences within the IWARS.

15. SUBJECT TERMS
BEHAVIOR, COGNITION, DECISION MAKING, INTEGRATED SYSTEMS, SOFTWARE, PERCEPTION, PROBLEM SOLVING, DISMOUNTED SOLDIER, VARIABLES, SIMULATION, SMALL UNIT OPERATIONS, COMPUTERIZED SIMULATION, INFERENCE, SBIR REPORTS, INFORMATION PROCESSING, IWARS(INTEGRATED WARFARE SYSTEMS)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dean-Michael Sutherland
U	U	U	SAR	28	19b. TELEPHONE NUMBER *(Include area code)* 508-233-5558

Table of Contents

List of Figures

Preface

The U.S. Army Natick Soldier Systems Center develops and applies modeling and simulation (M&S) technologies to evaluate new warfighter systems in terms of their ability to maximize individual combatant (IC) effectiveness and small-unit warfighting capabilities. Currently, the behavior models that control the warfighters in these M&S efforts are fairly simple, relying predominantly on behavior scripting and simple rules to produce actions. As a result, the simulated entity does not reflect much of the critical inference, situation awareness, and decision making required by actual soldiers operating in the combat environment and, as such, degrades the realism of the models and therefore any conclusions drawn from the simulation-based studies. For example, the Infantry Warrior Simulation (IWARS), a constructive simulator of dismounted infantry, provides pre-defined transition possibilities between simulated behaviors, but does not allow those transitions to happen based on the kinds of inferences that human soldiers actually use when understanding their situation and deciding what to do.

This report discusses an effort to develop and implement an easy-to-use "Perceptual Editor" and to characterize the performance of two different mechanisms for transforming ground-truth to support these human-inference modeling needs. The work was carried out by Micro Analysis and Design of Boulder, CO 80301 under contract number W911QY-06-C-0005, during the period November 2005 – May 2006 under the sponsorship of the U.S. Army Soldier Systems Center, Natick, MA 01760.

Executive Summary

Where human behavior is often thought of in terms of a perception-action cycle, rich with interdependencies and fuzzy boundaries between processes, the human behavior representations of a computer-generated force implement a one-way process where exact, unambiguous data drive discrete, monolithic models of decision-making. The objective of this Phase I Small Business Innovation Research (SBIR) effort was to imbue human behavior representations with some of the complexities that characterize human perception and action. In more specific terms, Micro Analysis and Design's Phase I effort revolved around two related objectives. The first was to design and prototype a "Perceptual Editor" that could be integrated within the Infantry Warrior Simulation (IWARS) environment and would allow a user to define different kinds of transformations between ground truth variables and agent-based perceptions. The second was to investigate examples of the kinds of transformations that might be implemented using such an editor. Progress toward both of these objectives is described in this report.

Before developing the prototype "Perceptual Editor," we had to explore two interrelated sets of issues. The first had to do with identifying the appropriate levels of abstraction and aggregation within the IWARS environment where we might represent perceptual transformations. For example, "perception" for the computer-generated entities in a synthetic environment like the IWARS occurs at a very high level of abstraction; the objects of perception are given by well-defined data structures that characterize location, object type and other effective attributes. Even though such representations are far removed from the vagaries of raw perception (e.g., feature extraction within the visual field), they still leave plenty of room for other kinds of transformations that can influence entity behavior (e.g., shifting judgments of threat versus distance on the basis of experience with a given enemy). The issues here were to identify classes of entity perceptions that were not so abstract that they would obviate the need for perceptual transformation in the first place while, at the same time, supporting transformations at a sufficiently high level that they might still influence entity behavior. The second set of issues had to do with identifying the specific software mechanisms that would allow such transformations to be specified and realized within the IWARS environment. Here we needed to satisfy a set of requirements for software that was flexible and extensible and yet sufficiently self-contained to facilitate the verification and validation of the entity behaviors affected by such transformations while minimizing the modification to the IWARS code base itself. Our approach to both sets of issues was to describe how a "Perceptual Editor" might dovetail with the IWARS "Condition Wizard" user interface. In this way we not only fixed the level of abstraction at which perceptual transformation might be represented but we also located a natural software joint for integrating the perceptual editor into the IWARS code base.

Development of the Perceptual Editor itself resulted in a functional, stand-alone prototype capable of:

- Emulating input and output streams from a notional IWARS environment.
- Supporting user-defined mappings between ground truth data (input to the Perceptual Editor) and entity perceptions (output to the IWARS) via a variety of different transformation mechanisms.
- Supporting native calls to design and run-time environments for defining and implementing perceptual inferences. Such environments might be a straightforward as

formula definitions within an Excel® spread sheet, to the development of a full-blown cognitive model within the ACT-R architecture.
- Implementing an XML-file based communication between applications.

The second major thrust of our Phase I effort was the investigation of cognitively-inspired mechanisms for producing perceptual transformations. Whereas the Perceptual Editor provides an extensible framework for specifying and integrating such mechanisms within the IWARS, it is still up to the user to specify how a given transformation will occur. We looked at two potential kinds of transformations, keeping in mind how easily each could be invoked from the perceptual editor. First, we looked at how perceptions of distance to an enemy might be transformed into judgments of threat using an experienced-based model of an agent's long-term memory. These investigations yielded interesting results suggesting how agent perception in a simulated environment might be influenced by the statistical structure of the environment. In particular, we saw preliminary indications of how an agent's learned ability to judge threat is affected by increasing noise (i.e., statistical variability) in his environment and we outlined a path for a "poor man's" representation of signal detection in such environments. Second, we reviewed ongoing research in the ACT-R community for modeling an agent's ability to group objects into more meaningful wholes. In connection to the IWARS, the ability for an agent to infer the presence of fire teams, squads etc., from the perception of individual members would be useful. The ACT-R models demonstrate good fits to human performance on similar tasks and the ACT-R architecture itself provides many validated parameters that can be tuned to produce individual differences. In both cases, the cognitive mechanisms we considered provided ready representations of perceptual transformations potentially useful to the IWARS community. Moreover, these mechanisms demonstrate how the variability that characterizes so much of perception can be produced by straightforward representations of environments and individuals entities.

Finally, from a methodological point of view, we note the extent to which the demand for modular software and tools that facilitate the verification and validations of perceptual inferences have shaped our effort. Although we might speak of "integrating" perceptual inferences within the IWARS, the architecture we developed really supports more of an interleaving of inferences, in which both the design-time specification and run-time invocation of a perceptual inference would be handled by applications sitting outside of the IWARS. In this way, we not only maintain the integrity of the respective code bases, but we also circumscribe very clearly at a conceptual level what the interleaved applications are responsible for delivering.

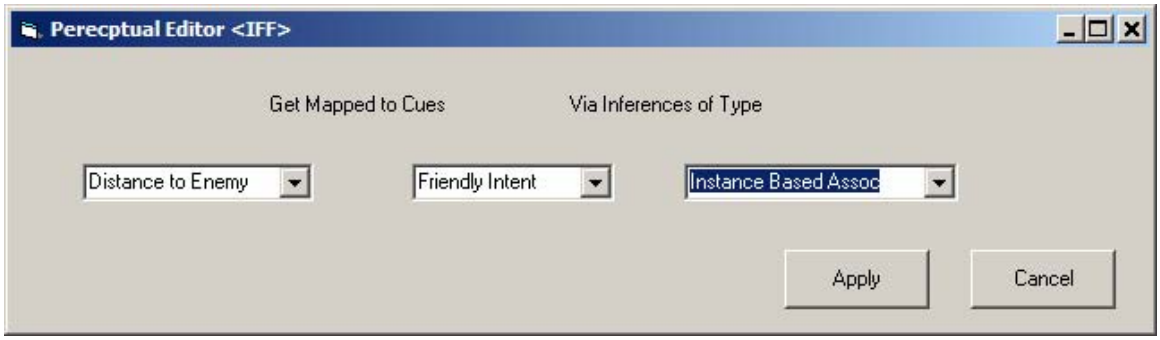
AGENT-BASED MODELING OF DISMOUNTED INFANTRY THROUGH INCLUSION OF PERCEPTIONS, INFERENCES AND ASSOCIATIONS – PHASE I

Introduction

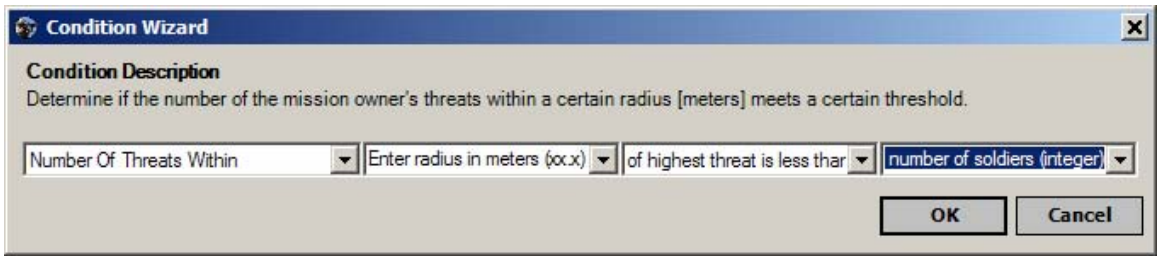
Where human behavior is often thought of in terms of a perception-action cycle, rich with interdependencies and fuzzy boundaries between processes, the human behavior representations of a computer generated force implement a one-way process where exact, unambiguous data drive discrete, monolithic models of decision making. In the most general terms, the objective of this Phase I SBIR effort was to imbue human behavior representations with some of the complexities that characterize human perception and action. In more specific terms, our Phase I effort revolved around two related efforts. The first was to design and prototype a "Perceptual Editor" that could be integrated within the Infantry Warrior Simulation (IWARS) environment and would allow a user to define different kinds of transformations between ground truth variables and agent-based perceptions. The second was to investigate examples of the kinds of transformations that might be implemented using such an editor. We looked at an agent-based mechanism for both inferring intent from cues within a "noisy" (i.e., stochastic) environment and as means for representing individual difference in "signal detection" (i.e., variation between agents in discerning changes in a noisy cue). In addition, we reviewed extant research on "projection" and perceptual grouping in which computational cognitive models were developed to reason about tasks in the same way that humans do. This report will describe these efforts below.

Developing a Prototype for a "Perceptual Editor"

In preparing for our kick-off meeting we developed a rough outline for a "Perceptual Editor" that could be accessed by an IWARS simulation developer. This outline took the form of a screen shot that we used to discuss the level of detail that would be needed for an IWARS user to define a mapping from simulation variables (i.e., ground truth) to perceptual cues via an inference mechanism.



Moreover, during our kick-off meeting, we identified the Condition Wizard within the IWARS development environment as a possible integration point for a perceptual editor. The Condition Wizard allows the user to define conditions that control tactical execution within the simulation. Essentially, this wizard allows the user to define the left-hand side (the condition portion) of an if-then rule.



The similarity of the look and feel between the two interfaces, though entirely unintentional, was a happy coincidence. During our discussion at the kick-off meeting, it became clear that a Perceptual Editor would essentially expand the left-most columns of the Conditions Wizard thereby allowing the user to define rules that map variables to cues via inferences. The inferences themselves might be simple if-then rules, or complex cognitive models of perception. The Perceptual Editor would allow the IWARS user to develop conditions for the conditions, so to speak, allowing for additional complexity at the "perceptual" level without having to modify behavioral libraries within the IWARS code base.

Having fixed the conceptual level at which the Perceptual Editor would be deployed within the IWARS development environment, we then started thinking about software integration. Although the specifics would depend on collaboration with the IWARS development team during a Phase II effort, we verified that a file-based interaction would be workable. Accordingly, our prototype was designed to read in both a list of available simulation variables (i.e., ground-truth) as well as a list of corresponding "perceptions"—i.e., the Boolean conditions defined through the Conditions Wizard. Depending on the complexity of the desired transformation from cue to "perception," we determined that it would usually be necessary to output a file from the Perceptual Editor. (We also considered that some transformation might be specified directly in the perceptual editor.) That file would reflect the user's initial mapping between variables and cues in a standard format to be read into whatever external application is being used to effect the transformation (e.g., allowing a user to define the instance-based long term memory structure that represents an agent's experience associating cues to perceptions).

The result of these discussions was a functioning software prototype for a Perceptual Editor. The prototype software included the following executables (.exe) and dynamic-link libraries (.dll):

NatickHarness.exe
NatickLibrary.dll
NatickInternalSupport.dll
EngineVariableAssociation.dll
ExcelEngineSetup.exe
ExcelEngineRuntime.exe

The NatickHarness.exe implements a connection between an emulated IWARS environment and various external perceptual inference engines. These functions are accessed by way of the top-level dialog depicted in Figure 1. This emulation is achieved using utilities that allow us to specify ground truth variables and "perceived" variables within IWARS as if the variables had been given in a formatted file (e.g., XML or CSV). Similarly, an interface to an external

inference engine is provided for both authoring and execution—in this case a simple Excel spreadsheet. Example mechanisms for launching the Engine/Variable association dialog, specifying Ground Truth values, and executing external simulation Engines are also provided.

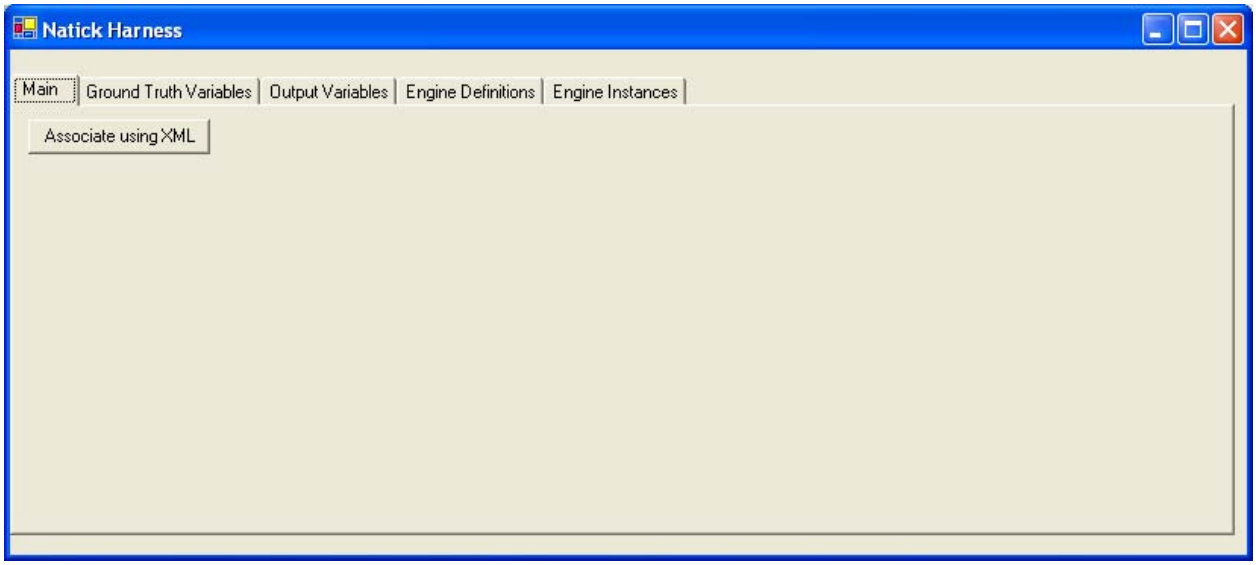


Figure 1. The Top-Level User Interface for the Perceptual Editor

The NatickLibrary.dll provides methods for launching the Engine/Variable association dialog, setting and retrieving ground truth and "perceived" variable values, specifying external simulation Engine definitions, and generating and using instances of engine/variable associations (Engine Instances). This library is used to abstract the internals of this implementation of XML or CSV communication from the NatickHarness.exe (or, conceptually, an IWARS-like tool).

The NatickInternalSupport.dll provides classes and methods supporting XML description of ground truth variables, "perceived" variables, Engine Definitions and Engine Instances (associations between ground truth variables, output variables and a particular Engine type). These descriptions are given using the interfaces depicted in Figures 2 and 3.

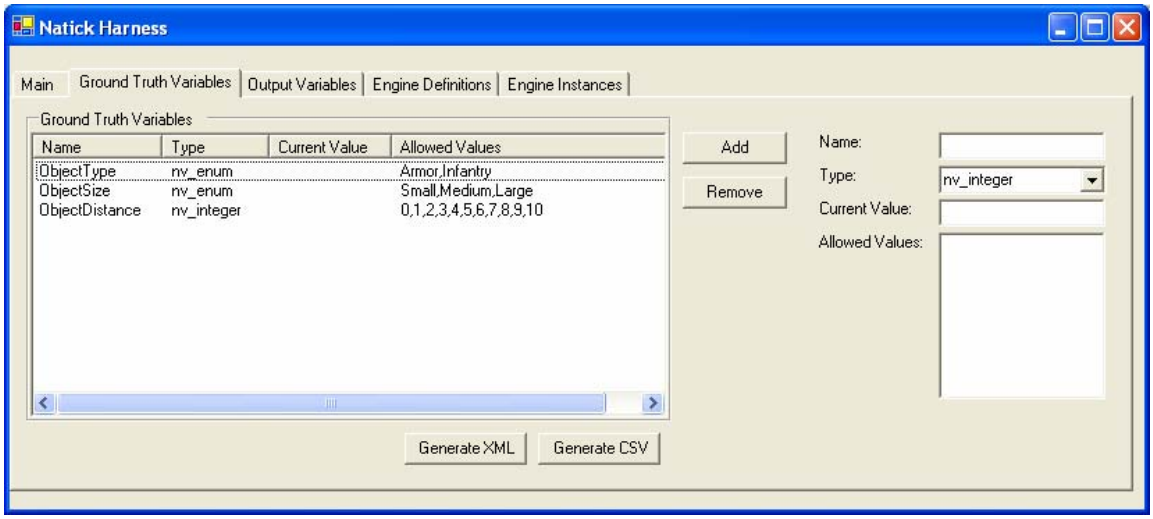


Figure 2. An Interface for Defining Ground Truth Variables (as they might be exported from the IWARS)

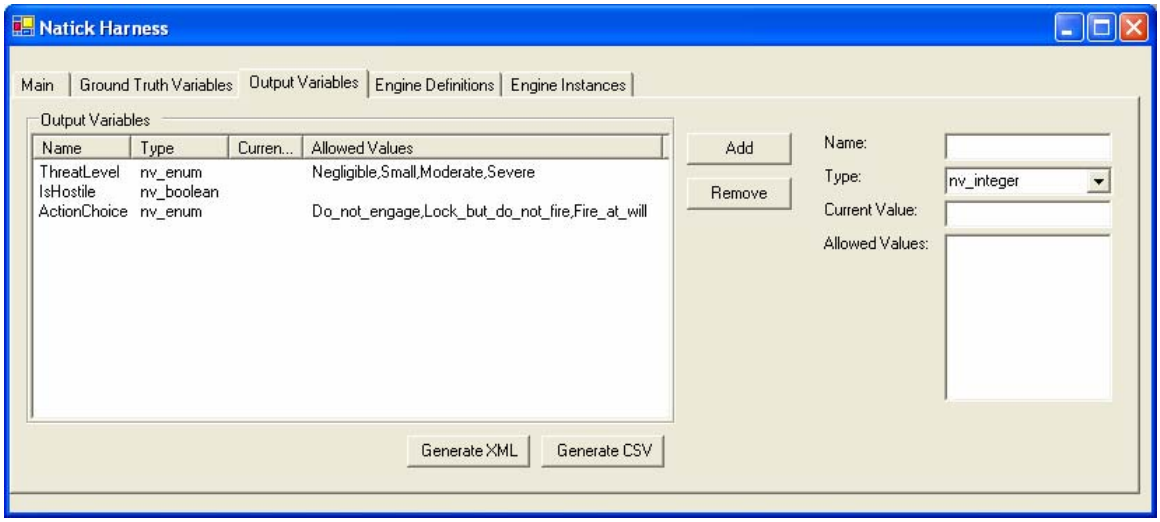


Figure 3. An Interface for Defining "Perceived" Variables (i.e., the results of the perceptual transformations that are mapped back onto variables within the IWARS)

The EngineVariableAssociation.dll provides a dialog for associating ground truth and "perceived" variables with an Engine Definition to produce a specific Engine Instance (the dialog is depicted in Figure 4). The Engine Instance can then be used to stimulate "perceived" variable states based on particular ground truth variable states. The dialog offers facilities to call the Setup executable for the particular Engine Definition and prepare how the "perceived" variables will be stimulated based on ground truth (input) states. The dialog also allows the user to save the list of defined Engine Instances (they are saved as an XML file).

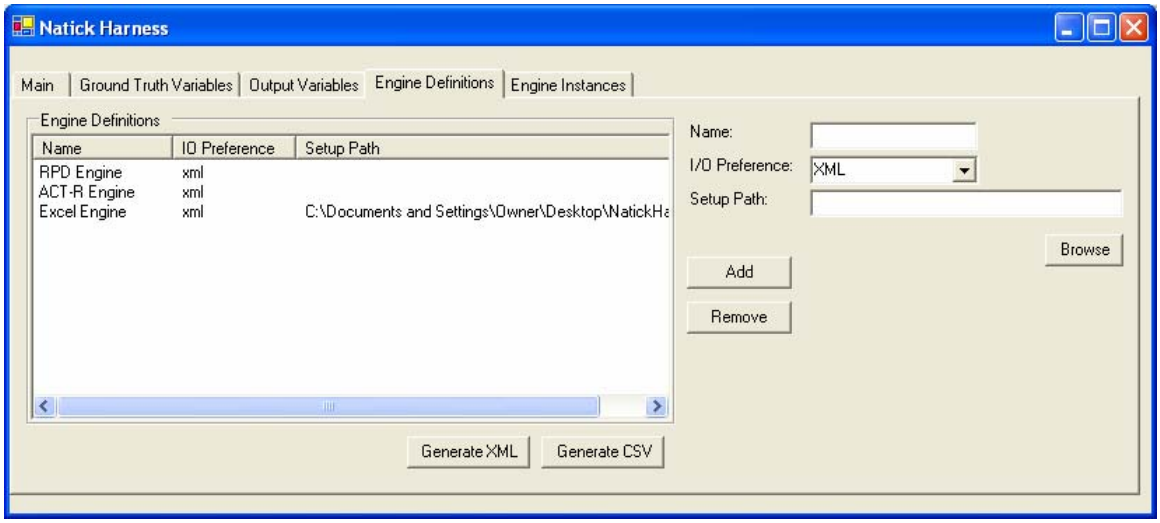


Figure 4. An Interface for Associating Mappings to an Inference Engine

In this case the "Inference" will be implemented by way of an excel spreadsheet using the Interface in Figure 5.

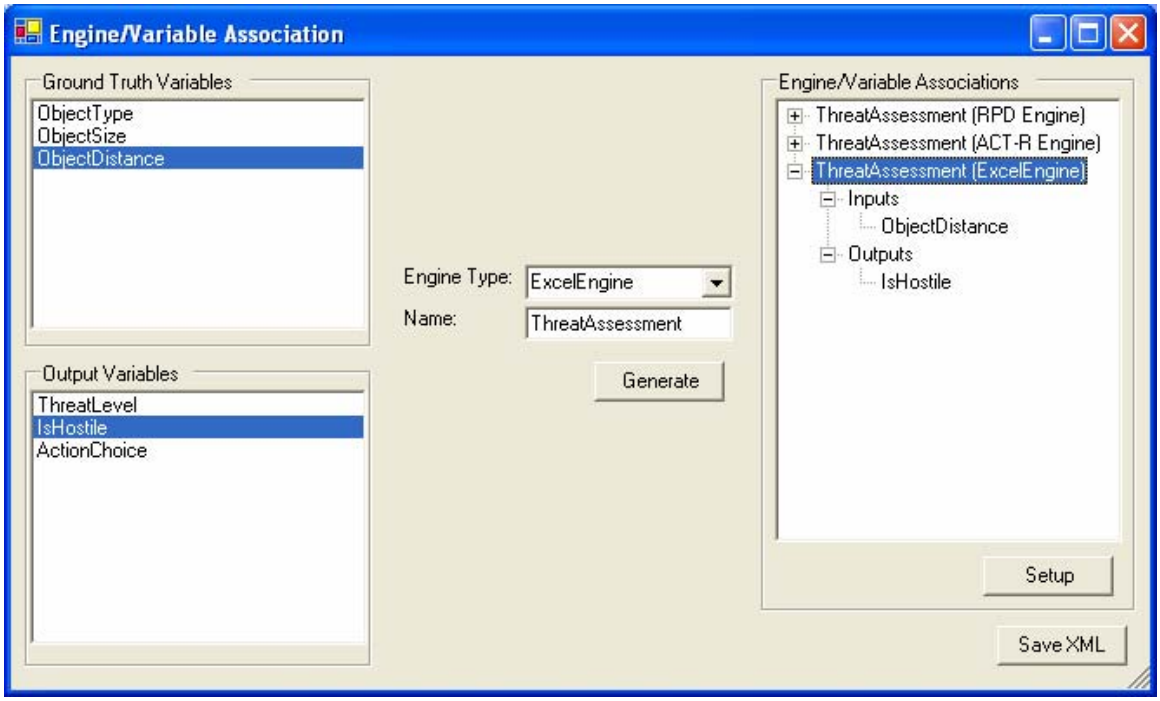


Figure 5. An Interface for Defining a Particular Instance of an Inference Engine

In this case, by hitting "Set-up" the user opens an excel spreadsheet populated with the various ground truth (input) and "perceived" (output) variables to be associated using whatever formula the user defines.

The ExcelEngineSetup.exe provides one example of how an Excel Engine might be used as an inference engine. It allows the user to set output states based on input states (and possibly other user defined elements) for any particular Excel Engine Instance.

A variety of XML files are used and produced by the Natick Tool. They are as follows:

groundtruth.xml
outputvariable.xml
enginedef.xml
engineInstances.xml

Each file contains a description of the area indicated by the name of the file (e.g., groundtruth.xml describes the list of available ground truth variables available in the Natick Harness, or, conceptually, in IWARS).

This is an example of a ground truth XML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<GroundTruthVariables>

  <GroundTruthVariable Name="ObjectType" Type="nv_enum" CurrentValue=""
  AllowedValues="Armor,Infantry" />

  <GroundTruthVariable Name="ObjectSize" Type="nv_enum" CurrentValue=""
  AllowedValues="Small,Medium,Large" />

  <GroundTruthVariable Name="ObjectDistance" Type="nv_integer" CurrentValue=""
  AllowedValues="0,1,2,3,4,5,6,7,8,9,10" />

</GroundTruthVariables>
```

Each ground truth variable has attributes for its name, type, current value and allowed values. The name can be a text string and should be suitable for display to a user. The type can be one of:

nv_enum
nv_integer
nv_boolean
nv_real
nv_string

The current value is a string representation of the current value of the described variable (it is optional and may be missing or may be an empty string to indicate “no current value”). The allowed values are a comma separated string of values the variable is allowed to take (it is also optional and may be missing or may be an empty string to indicate “no limitations within variable type”). Note that, currently, no validity checks are done on current value to ensure it falls within the allowed values. There is also no checking done to ensure that enumerations have a list of allowed values.

The outputvariable.xml file, which describes the “perceived” variables, follows an identical format to that of the ground truth file.

This is an example of an engine definition XML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<EngineDefinitions>
  <EngineDefinition Name="RPD Engine" IOPref="xml" SetupPath="" RuntimePath="" />

  <EngineDefinition Name="ACT-R Engine" IOPref="xml" SetupPath="" RuntimePath="" />

  <EngineDefinition Name="ExcelEngine" IOPref="xml"
  SetupPath=".\ExcelEngineSetup.exe" RuntimePath=".\ExcelEngineRuntime.exe" />

</EngineDefinitions>
```

The name can be a text string and should be suitable for display to a user. The IOPref indicates the file type the engine uses for communication and must be either XML or CSV in the Natick Harness interface. This could be expanded to provide other forms of communication between the Natick Harness (or, conceptually, IWARS) and the various simulation engines. Currently, the harness only supports XML file communication.

The SetupPath and RuntimePath are (possibly relative) paths to the executable(s) responsible for engine variable setup and runtime stimulation of variables.

This is an example of an engine instance XML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<EngineInstances>
  <EngineInstance Name="ThreatAssesment" Type="RPD Engine" CallingSequence="">
        <Inputs>
              <Input Name="ObjectDistance" Type="nv_integer" CurrentValue=""
  AllowedValues="0,1,2,3,4,5,6,7,8,9,10" />
        </Inputs>
        <Outputs>
              <Output Name="IsHostile" Type="nv_boolean" CurrentValue=""
  AllowedValues="" />
        </Outputs>
  </EngineInstance>

  <EngineInstance Name="ThreatAssesment" Type="ACT-R Engine" CallingSequence="">
        <Inputs>
              <Input Name="ObjectDistance" Type="nv_integer" CurrentValue=""
  AllowedValues="0,1,2,3,4,5,6,7,8,9,10" />
        </Inputs>
        <Outputs>
              <Output Name="IsHostile" Type="nv_boolean" CurrentValue=""
  AllowedValues="" />
        </Outputs>
  </EngineInstance>

  <EngineInstance Name="ThreatAssesment" Type="ExcelEngine"
  CallingSequence=".\ThreatAssesment_ExcelEngine.xls">
        <Inputs>
```

```
            <Input Name="ObjectDistance" Type="nv_integer" CurrentValue=""
        AllowedValues="0,1,2,3,4,5,6,7,8,9,10" />
        </Inputs>
        <Outputs>
            <Output Name="IsHostile" Type="nv_boolean" CurrentValue=""
  AllowedValues="" />
        </Outputs>
  </EngineInstance>
</EngineInstances>
```

The name can be a text string and should be suitable for display to a user. The type must be one of the names listed in the engine definitions in the enginedef.xml file. This is not enforced in the XML file, but is enforced in the Natick Harness (or, actually, the Engine/Variable association .dll; only defined engines are available for instantiation).

The calling sequence contains information specific to how the runtime inference engine should be called (e.g., for the ExcelEngine, the ExcelEngineRuntime is called with the Excel file containing the relevant formulas (i.e., "perceptual inferences") and variable values as "ExcelEngineRuntime.exe –f<calling sequence string>"; this same mechanism can be used for other inference engine types).

The Inputs and Outputs are reproductions of the variable information associated with this engine type during Engine/Variable association. The format of this data is identical to that described in the ground truth XML file description.

Currently, the only calling sequence supported is "<Engine Runtime Executable> -f<File generated during Engine Setup>". This could be modified to support many other possible calling sequences.

Exploring Perceptual "Inferences"

The Perceptual Editor provides an extensible framework for defining and integrating perceptual inferences within the IWARS. Although it places constraints on the manner in which such inferences are integrated (by standardizing the input and output between the IWARS and an inference engine), the Perceptual Editor does not affect the inferences themselves. That is, even given the input and output from such an inference engine, it is still up to the user to specify whatever mechanism will be required to perform the inference. At a minimum, a user might simply describe a mathematical function that relates cues to perceptions, for instances, using an Excel spreadsheet as in the previous example. However, there are also cases where the user might want to take advantage of inferences implicit in the more complex workings of a cognitive model. (Even if all computational models could be regarded as computing *some* function, it is often more convenient to view the transformation as mediated by some theoretical constructs rather than as a disembodied mathematical function).

Our goal in Phase I was not to canvas all the possible cognitive mechanisms for perceptual inference, nor was it to advocate a particular cognitive architecture for the job. Instead, we identified a few perceptual inferences with obvious relevance to infantry behavior and explored how they might be realized using familiar approaches (familiar to us, at least). We had several

goals in mind for this exercise. The first goal was to provide concrete examples of what we mean by the term "perceptual inference" and to demonstrate by way of these examples how the fidelity of a simulation environment like the IWARS might be extended. The second goal was to mark out a clear division of labor between the perceptual and the behavioral and determine the extent to which this division could be supported within the Perceptual Editor. Finally, the third goal was to work through an (almost) end-to-end example of specifying a perceptual inference with an eye toward a use case and, eventually, a methodology for integrating such inferences. We first describe the inferences themselves and then we discuss how well we met our three goals and the other overarching goals of this project.

An Instance-Based Perception of Distance[*]

The representation of distance is bedrock in any computer simulation of military activity, and it figures prominently in determining entity behaviors as both physical objects (e.g., a target that can be hit at some distance with a given weapon) and as agents (e.g., a commander who must decide whether another entity poses a threat). Apart from the "physics" of the simulation, where distance calculations are as exact as they can be, the representation of distance as a driver in human behavior representations should be far more subjective and context-sensitive than it currently is. For example, in reality, a fighter pilot has a far different sense of distance than a dismounted infantry soldier. While this difference might be reflected in a simulation by using different agent models (pilot vs. dismounted soldier), there are cases in which the perceptions of even a single agent might be context sensitive. For example, an individual dismounted soldier might have a different sense of distance depending on whether he is engaging the enemy in open terrain or, instead, in the complex interior spaces of a MOUT environment. In such cases, crisp rules that transform distance given by some fixed metric (e.g., meters) are not likely to reflect these different, context sensitive perceptions of distance.

To explore an alternative to such rule-based approaches, we developed a deliberately simplistic stand-alone model that transforms distance, represented as ground truth, into a perception of threat by way of an instance-based agent model developed at MA&D (see Warwick et al. 2001 for details about the instance-based architecture). In this case, we actually specified two transformations, one explicitly and the other implicitly. The first transformation was the explicit "binning" of a real-valued distance variable into a set of discrete values that would be presented to the agent. At this point it would have been possible to produce a variety of mathematical transformations between actual and "perceived" distance by using clever mappings (e.g., picking non-uniform bin sizes). Instead, we took a straightforward approach in which we mapped real-valued ranges of equal "width" onto enumerated values representing distance. The second transformation occurred implicitly within the instance-based architecture as the agent learned the distance at which the enemy would become threatening. This point is crucial; rather than represent this threshold as a crisp rule internal to the agent (e.g., if distance < 5, then threat = high), the transformation from distance to threat was the product of an environment that was both variable and noisy. Thus the threshold at which an enemy would become threatening would

[*] The executable models described in this section are available for download and inspection at: http://www.maad.com/index.pl/brims_models

change from simulation to simulation (or even within a simulation), while the threshold itself represented only the average distance at which the enemy was likely to become threatening.

During a simulation run, the agent would "see" an enemy at some distance and then "decide" whether the enemy was threatening, receive feedback about the correctness of the decision and then repeat with a new enemy at another distance. In this way, the agent-based perception of threat amounted to an experience-based estimate of the actual distance at which the enemy would become threatening. Because that threshold was defined according to a distribution type, with a mean and standard deviation, distance as a cue could be more or less informative (i.e., predictive of enemy threat). Note also that the threshold value itself could change, so that past experience with one "population" of enemy might adversely influence future perception. To demonstrate these vagaries of perception, we ran the model under a variety of circumstances (e.g., different mean values for the distance at which a given enemy "population" becomes threatening, different amounts of variability about those mean values, etc.). Although we have yet to summarize data across a set of models, the results of the individual model runs depicted (Figures 6 and 7) suggested that the agent-based approach is capable of supporting a more nuanced perception of distance.

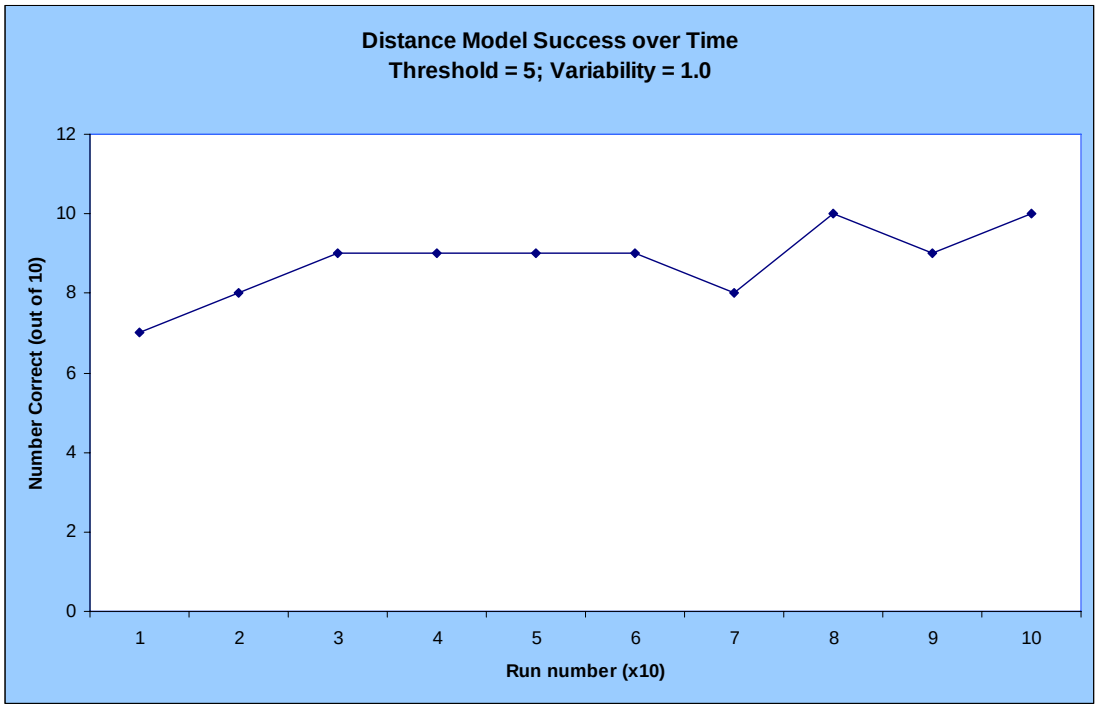


Figure 6. Performance of a single model run over time (i.e., repeated decision trails) as it learns the distance at which a given enemy population becomes threatening

"Number Correct" reflects the correct identification of both threatening and non-threatening enemy.

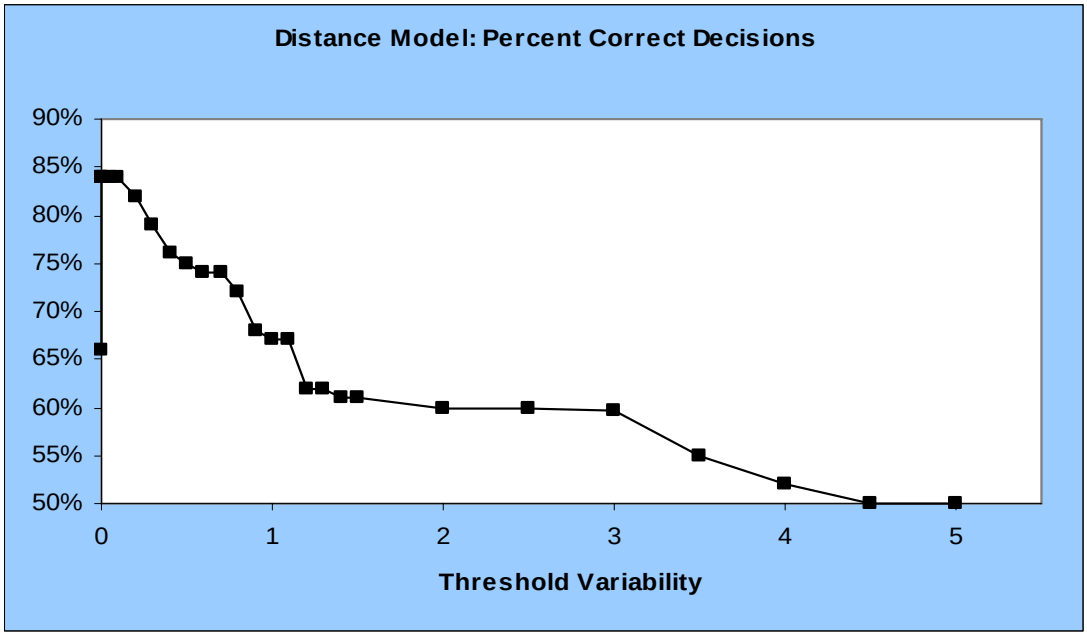


Figure 7. The effect of Variability on the Model's Ability to Recognize the Distance at which the Enemy becomes Threatening

Note that as variability about the threshold increases, distance becomes less predictive as a cue and the model's performance diminishes.

Although seemingly straightforward, this experienced-based transformation of distance introduces potentially interesting and useful interactions in agent-based perception. Chief among these is that the transformation itself is a reflection of the statistical structure of a (possibly noisy or unstable) environment rather than the strict application of a cut-off rule that has been specified *a-priori*. Thus the agent's perception reflects as much uncertainty as is warranted by the environment. In those cases where variability in the enemy behavior is low, the agent's perception is relatively certain (as reflected in the model's performance correctly perceiving the enemy as threatening or not); in those cases where the variability is high, the agent's perception is less certain. Moreover, a single agent model can be deployed under any of these conditions and will adapt to the structure of the environment. That is, the agent's perceptions are truly a product of the environment rather than the result of fine-tuning internal model parameters. In those cases where it is important to simulate adaptive behavior, this level of fidelity in the representation of agent perception would not be easily achieved even using a fuzzy rule.

Toward an Instance-Based Model of Signal Detection[*]

In addition to providing a potentially useful level of adaptive behavior, this experienced-based model of distance perception led us to consider how the same models might be used to represent an agent's ability to discriminate subtle changes in a given cue. Our sense of "signal detection" here is a bit metaphorical, but the idea is that given a "noisy" cue (like the distance cue described previously), some agents will be able to discriminate meaningful swings in a cue's value from those that are merely the product of random variability. Although we approached this problem

[*] The executable models described in this section are available for download, inspection at: http://www.maad.com/index.pl/brims_models

using the experienced-based models described earlier, we were not so much interested in the fact that such models are adaptive but, rather, that different learning regimens might ultimately lead to "individual differences" among agent models to discriminate the values of noisy cues. As before, we were trying to exploit the potentially subtle interactions here between the statistical structure of the environment and the parameters that affect the agent's learning performance.

To determine whether we could represent such individual differences, we first looked at a model's ability to recognize when a threshold had changed during a simulation (e.g., enemy that were threatening only at 5 units are now threatening at 2 units). Of course, the real question is one of how small a change can the model detect, which in turn, depends on the variability of the threshold. Again, we have yet to summarize data across a set of models, but the results of the individual model runs depicted suggest that the models are sensitive to changes in threshold values (Figure 8) and that the interaction between the size of the change and the amount of variability in the cue (i.e., signal to noise) is reflected in a plausible way on the model's performance (Figure 9).

Although these results are preliminary and will require additional analysis before we can be sure they are robust, they are suggestive of the kinds of agent-based performance we'd like to engender with a perceptual inference. The perception of even a ubiquitous cue like distance is neither monolithic, nor uniform, but depends on the environment in which it is presented. In a real sense, the meaning of a cue is to be found in the environment and not in the agent's head, so to speak, in the form of a hard and fast rule. At the same time, we're still interested in determining whether we can represent those differences inside the head that influence how well an agent is able to attune his perceptions to a noisy environment. We didn't complete those studies during our Phase I effort, but we are hopeful that straightforward parameter variations in the instance-based agent model will allow us to generate a variety of different performance curves of the kind depicted in Figure 9. We plan to pursue this work under another ongoing research contract.

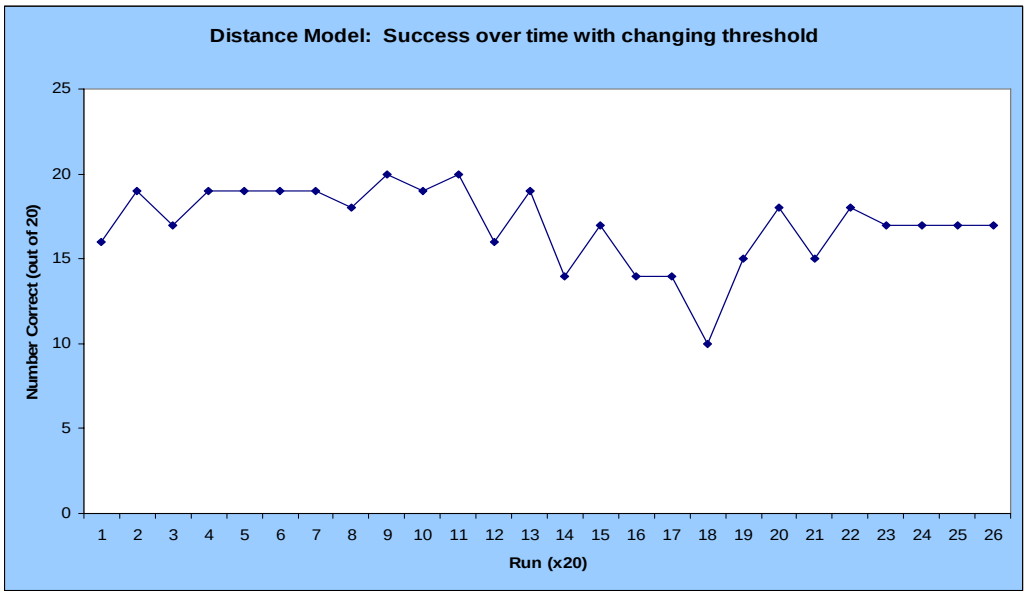


Figure 8. The graph depicts the results of a single model run where the distance threshold changed from 5 units to 2 half way through the run (with variability held constant)

The model falters after the switch—after consistent, near perfect performance, performance becomes volatile and diminishes to near chance levels. But the model is ultimately able to adapt to the change, returning to more consistent, albeit slightly less than perfect performance. This suggests that the new threshold has, in fact, been recognized (otherwise, we'd expect performance to remain low).

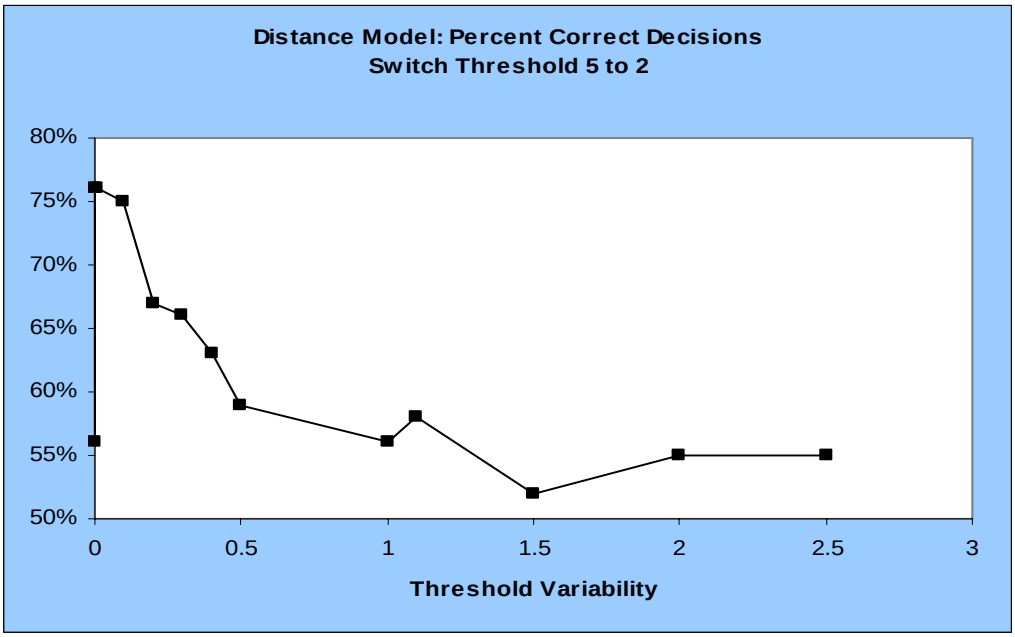


Figure 9. This graph shows how increasing variability affects the models ability to discriminate new threshold values from old, and mirrors the performance of the model in stable environments, as depicted in Figure 7

As variability (i.e., noise) increases the model's performance during any one simulation diminishes, suggesting that the change in threshold values has been lost in the noise.

Using ACT-R as a Perceptual Inference Engine

In addition to our in-house familiarity with instance-based agent models, we also exploited in-house expertise with the ACT-R cognitive architecture (Anderson and Lebiere 1998) to explore additional kinds of perceptual inference. While ACT-R has implemented a theory of perception by way of its perceptual motor module, the theory is couched at a lower-level of analysis. For example, the perceptual motor module is used to represent perceptual "bottlenecks" in which various "channels" of perception compete for attention and also to represent the agent's ability to identify and remember locations within the visual field. Our interest was at a higher, cognitive level that is better suited to describing how "meaning" might be inferred from the raw perception of cues. Thus we focused our informal discussions on two general sorts of mechanisms: the "mathematical" ability to project cue values and the ability to aggregate individual entities into groups.

The idea for a mathematical approach to perceptual inference followed from earlier independent efforts to develop cognitive models of a human interpreting a graph (e.g., a graph of supply vs.

demand). In that domain, the human (and, hence, the cognitive model) must be able to project trends, identify intercepts and intersection points in order to understand the graph. Although much of this depends on the interpretation of the graph qua image, the cognitive modes also used higher-level representations of the mathematics depicted in the graph in order to extract "meaning." For example, projecting past the intersection point of a supply and demand graph allows the agent to infer how an increase in price might affect demand. We discussed whether a similar approach could be taken in the context of perceptual inference where, for example, the future location of an object might be inferred from a cue giving current location and another cue indicating velocity. Couched at this level, a perception of location would more directly support complex inferences about relative location of a target ahead or behind (rather than constantly re-calculating relative position anew). Our preliminary discussions suggested that such an integration through our Perceptual Editor would be possible. More specifically, we canvassed the kinds of ground truth information that would be required to support such reasoning and we considered whether the perceptions so transformed could be mapped onto plausible variables within a military simulation environment like IWARS.

We also discussed how ACT-R might be used to infer groups given cues about the locations of individual entities. This mode of inference has more direct bearing on the kinds of inference that could be realized in the simulation of dismounted infantry. For example, the perception of a single entity will have different meaning to an agent depending on whether that entity happens to be a lieutenant, from which the presence of an entire platoon might be inferred, or a single member of a two-man rifle team. At a more basic level, and without invoking domain knowledge about military organization, the perception of multiple scattered entities will have different meanings depending on whether the presence of a single, dispersed group is inferred or, alternatively, whether the presence of multiple, partially represented groups is inferred. In either case, the perception of ground truth—i.e., individual entities and their locations—is transformed into more meaningful terms—i.e., groups and their centers of mass. Best and Gunzelmann (2005) describe a cognitive model that performs exactly these kinds of transformations; given inter-entity distances, these models aggregate entities to groups in much the same way that humans do (see Figure 10).

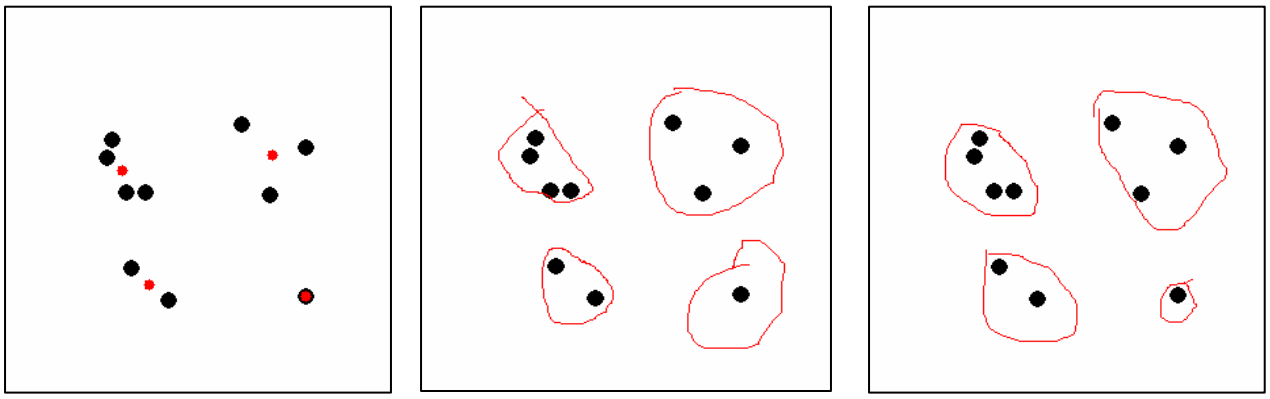


Figure 10. Entity groupings according to an ACT-R model and human subjects

Each pane depicts a set of entities to be grouped; the leftmost pane shows how entities were grouped by an ACT-R model, (with center of group mass indicated by the red dots); the middle and right pane show how the same entities were grouped by two different human subjects.

The model Best and Gunzelmann describe can be parameterized to produce individual differences in grouping strategy and, moreover, could be extended to reflect the influence of domain knowledge on perception of groups. Although we see a wide variety of potential applications for such inferences in an agent-based model, our preliminary discussions focused on the basic conceptual issues entailed by the integration of perceptual grouping algorithms through our Perceptual Editor (e.g., whether we could extract the right kinds of cues and return the "meaningful" results of an inference). Again, the integration required at this conceptual level would be relatively straightforward.

Finally, we talked about what would be required to support a native connection from the Perceptual Editor to an ACT-R modeling environment, both for specification of the cognitive transformation and the run-time connection to an inference engine. Though less interesting from a theoretical point of view, integration at this level would require some significant software engineering. Whereas the connection we implemented between the Perceptual Editor and an Excel spread sheet was straightforward, the ACT-R modeling environment and run-time engine are currently implemented in Lisp. This makes intercommunication to Windows environments either expensive, or tedious or both. Nevertheless, one of the design considerations behind our Perceptual Editor was the specification of a fairly generic file-based communications between the editor and the inference engine; this would serve to mitigate some of the integration risk.

Recommendations and Conclusions

As we indicated previously, the motivations for exploring specific inferences (i.e., distance perceptions and entity groupings) were threefold. First, we wanted concrete examples of what it might mean to specify a perceptual inference. Second, we wanted to define a clear division of labor between such inferences and the other kinds of agent behaviors in a simulation. Third, we wanted to work toward a use case both to identify gaps in our approach and to make sure that it was addressing some of the larger methodological concerns of the IWARS community. As it happens, these three motivations turned out to be somewhat interdependent.

Having a concrete example of a perceptual inference on the table is useful for all sorts of reasons, not the least of which is that the example serves to deflate rhetoric and disambiguate vocabulary. But on a deeper level, the examples allowed us to identify in fairly specific terms the input a Perceptual Editor would need to receive from the IWARS and the output it would return in order to effect a particular inference. (This was all the more important given that we were emulating data streams to and from the IWARS as we developed our prototype editor and we had to decide what to include in those emulated data streams.) So, for example, in discussing what it would mean to infer threat from distance, we identified distance-to-enemy as a necessary input to a perceptual inference, and a binary judgment of threat, no-threat as the output. Moreover, having pinned down the input and output, we could then focus on the various kinds of perceptual inference that could be specified for a single input-output pair. For example, using an Excel spread sheet to define a mathematical relationship between distance and threat yielded a straightforward rule-based view of perceptual inference, whereas using a naturalistic model allowed us to explore more subtle, learned behaviors. This is not to argue for one view of distance perception over the other but merely to point out that both would be possible using our Perceptual Editor. Likewise, the ACT-R models of grouping suggest another potential avenue for representing a perceptual inference. But in this case, the concrete example also exposed

specific software requirements, namely, the ability to integrated Lisp-based tools for both defining and running an ACR-R cognitive model within a Windows application (requirements we discussed but did not formally specify during Phase I). Ultimately, these concrete examples reveal the requirements for supporting an end-to-end process where an IWARS user would open the Perceptual Editor and specify both the input variables and inference mechanisms while minimizing the effort needed to set up the native environment for whatever mechanism is being invoked.

Although this kind of user support would lessen the burden on the IWARS user hoping to model perceptual inference using different mechanisms, it doesn't address an important conceptual question. Consider again our example of inferring threat from distance. Although we have used phrases like "transforming distance," and "inferring threat" as if they were synonymous, there is still a real question about the division of labor between perception and inference in the representation of an agent's behavior. Indeed, once we have inferred threat, it's a small step to action—so why not skip the middle step and simple act directly on the perception and implement the whole process in a single behavior library? For us, the answer is given essentially by fiat, insofar as we have designed the Perceptual Editor to hang off the Conditions Wizard rather than be hard-coded into the IWARS. Still, the conceptual questions remains, and we often found ourselves thinking about additional sorts of perceptual transformations that were not so easily disentangled from the actions they supported (threat assessment is a prime example). Conversely, we also found ourselves thinking about how ground truth might be uniformly transformed independent of any particular agent action or inference engine. For example, representing the limits of visual acuity would also serve to transform ground truth variables into more realistic perceptions, but such a transformation would apply across all agent behaviors. This led us to propose as part of a Phase II effort, additional functionality *within* the Perceptual Editor to allow a user to link directly to a handful of empirically derived models of sensory acuity. This impulse to link some transformations to specific behaviors and others to more global mechanisms only serves to blur the division of labor between the perception and inference on which we predicated the design of our Perceptual Editor.

Nevertheless, in confronting this fact, we did come to a deeper appreciation of some of the methodological concerns that were presented to us at the outset of the effort. Two concerns in particular stood out: that the integration of a perceptual inference within the IWARS should entail the least modification to the IWARS code base as possible and that the integration of any one inference should have a clear audit trail. We responded to these concerns by developing a modular approach built around the standards that would emerge from linking inferences by way of a single interface. As we indicated previously, we envisioned the Perceptual Editor as an adjunct component to the IWARS. Although we spoke of "integrating" perceptual inferences, the architecture we proposed really supports more of an interleaving of inferences, in which both the design-time specification and run-time invocation of a perceptual inference would be handled by applications sitting outside of the IWARS. Moreover, the brokering of design-time and run-time interactions would be mediated by specific file formats. In this way, we not only maintain the integrity of the respective code bases, but we also circumscribe very clearly at a conceptual level what the interleaved applications are responsible for delivering. This combination of interleaved applications and well defined input-output requirements supports a high degree of modularity while making it easier to inspect and verify the specifications for a particular

transformation in the very development environment used to specify the transformation. So, while this doesn't answer the conceptual question about the principled division of labor between the perceptual and inferential, it does make it much easier to isolate whatever it is that we happen to treat as perceptual by insulating the IWARS code from the mechanisms we use to define the transformation. Moreover, should we come to re-draw the line between the perceptual and inferential, our approach is by its nature easily extensible.

Our Phase I approach was focused on the development of software framework for specifying perceptual inferences. We also investigated specific inferences to demonstrate that even seemingly straightforward perceptions like distance or entity count might be represented in very different ways. Indeed, by supporting external representations of such inferences it is possible to choose the fidelity of representation (e.g., using a rule-based approach versus the instantiation of a cognitive model) and, hence, to bring a variety of different factors to bear on perceptual inference (e.g., influence of environmental factors, individual differences, principled variability etc) But, as with any tool, our framework is by itself inert, so the work remains to investigate additional inferences and identify those that are likely to have high payoff for the IWARS simulation community. That said, we believe our Phase I effort demonstrates the extent to which that research would be facilitated by adopting the proper framework.

This document reports research undertaken at the U.S. Army Research, Development and Engineering Command, Natick Soldier Center, Natick, MA., and has been assigned No. Natick/TR-07/004 in a series of reports approved for publication.

References

Anderson, J. R., & Lebiere, C. (1998). The Atomic Components of Thought. Mahwah, NJ: Lawrence Erlbaum Associates

Best, B. J. & Gunzelmann, G. (2005). Hierarchically-Based Perceptual Grouping in ACT-R. Presented at the Eleventh Annual ACT-R Workshop and Summer School.

Warwick, W., McIlwaine, S., Hutton, R. J. B., McDermott, P. (2001). Developing Computational Models of Recognition-Primed Decision Making. Proceedings for the Tenth Conference on Computer Generated Forces, Norfolk, VA, SISO.